AD-A106 802 MINNESOTA UNIV MINNEAPOLIS DEPT OF COMPUTER SCIENCE F/G 9/2
OPTIMAL BPC PERMUTATIONS ON A CUBE CONNECTED SIMD COMPUTER.(U)
AUG 81 D NASSIMI, S SAHNI N00014-80-C-0650
UNCLASSIFIED TR-81-28 NL

1 OF 1
AD
A106802

END
DATE
FILMED
12-81
DTIC

NR - 049-477'

12

Computer Science Department
136 Lind Hall
Institute of Technology
University of Minnesota
Minneapolis, Minnesota 55455

15 N00014-80-C-0650

14 TR-81-28

6 Optimal BPC Permutations On A
Cube Connected SIMD Computer.

by

10 David/Nassimi [redacted] Sartaj/Sahni

9 Technical Report, 81-28

11 Aug [redacted] 81

12 15

Cover design courtesy of Ruth and Jay Leavitt.

412021

Optimal BPC Permutations On A Cube Connected SIMD Computer

David Nassimi[+] and Sartaj Sahni[++]

Abstract

In this paper, we develop an algorithm to perform BPC permutations on a cube connected SIMD computer. The class of BPC permutations includes many of the frequently occurring permutations such as matrix transpose, vector reversal, bit shuffle, and perfect shuffle. Our algorithm is shown to be optimal in the sense that it uses the fewest possible number of unit routes to accomplish any BPC permutation.

Key Words and Phrases

BPC permutation, cube connected SIMD computer, complexity.

[+]Author's Address: Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, Illinois 60201

[++]Author's Address: Computer Science Department, University of Minnesota, Minneapolis, Minnesota 55455. The work of this author has been supported in part by the Office of Naval Research under contract N00014-80-C-0650.

1. Introduction

An SIMD (Single Instruction stream Single Data stream) computer is a parallel computer consisting of a large number of identical processing elements. A block diagram of such a computer is given in Figure 1. SIMD computers have the following characteristics:

(1) They consist of N processing elements (PEs). The PEs are indexed 0, 1,..., N-1 and an individual PE may be referenced as in PE(i). Each PE is capable of performing the standard arithmetic and logical operations. In addition, each PE knows its index.

(2) Each PE has some local memory.

(3) The PEs are synchronized and operate under the control of a single instruction stream. This instruction stream is generated by the control unit which has access to the program that is to be run.

(4) An enable/disable mask can be used to select a subset of the PEs that are to perform an instruction. Only the enabled PEs will perform the instruction. The remaining PEs will be idle. All enabled PEs execute the same instruction. The set of enabled PEs can change from instruction to instruction.

The essential feature that distinguishes one SIMD computer family from another is the interconnection network. In this paper, we are concerned only with two types of interconnection networks: the mesh and the cube.

(i) Mesh Connected Computer (MCC)
In this model, the PEs may be thought of as being logically arranged as in a two dimensional array $A(n_1,n_2)$ where $N = n_1 \times n_2$. The PE at location $A(i,j)$ is directly connected to the PEs at locations $A(i \pm 1,j)$ and $A(i,j \pm 1)$ (provided these PEs exist). Figure 2(a) shows the interconnections in a 4 × 4 MCC.

(ii) Cube Connected Computer (CCC)
Assume that the number of PEs, N, is a power of 2. So, $N = 2^q$. Let $i_{q-1} \ldots i_0$ be the binary representation of i, $i \in [0,N - 1]$ and let $i^{(b)}$ denote the number whose binary representation is

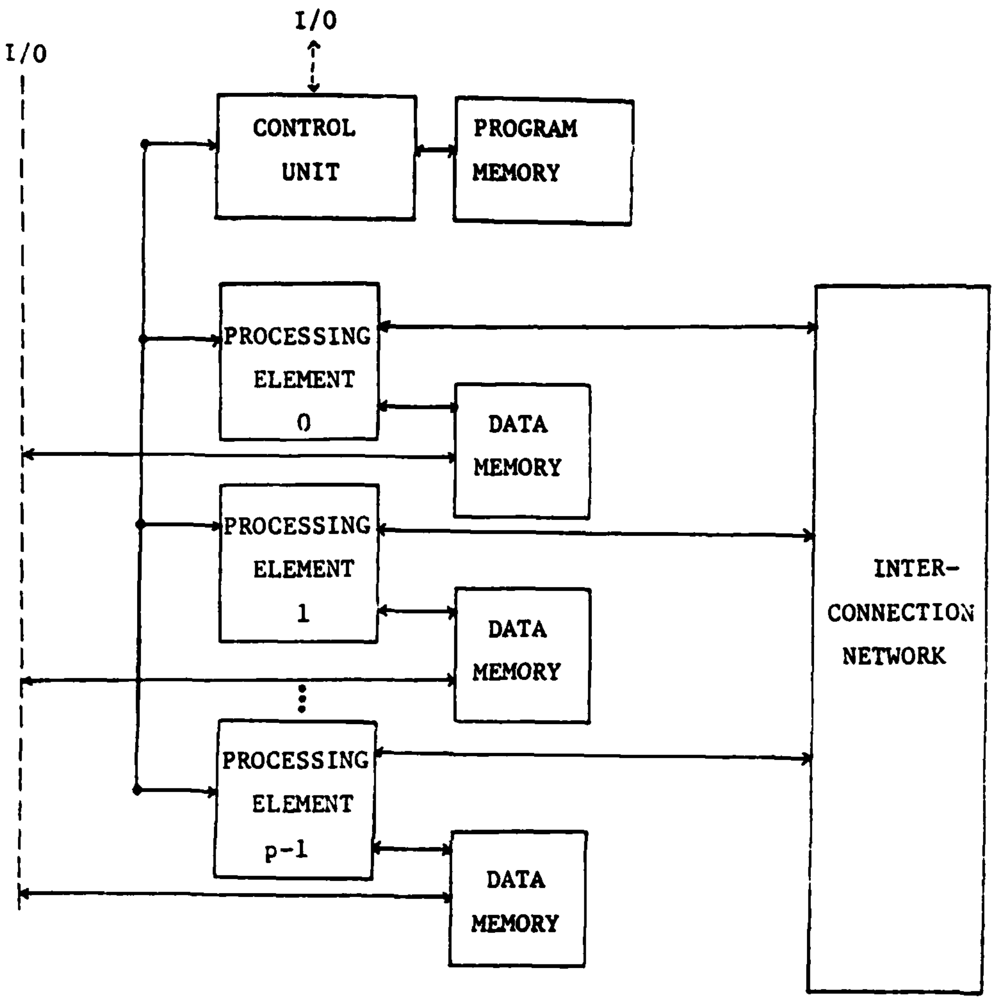
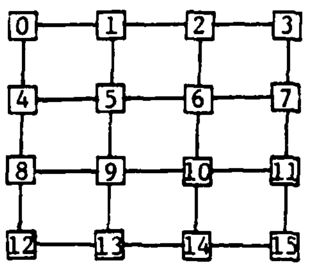


Figure 1 Block diagram of an SIMD computer

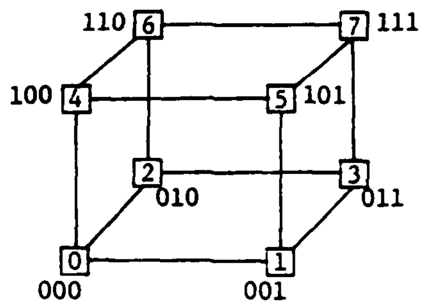
$i_{q-1}\ldots i_{b+1}\ \overline{i}_b\ i_{b-1}\ldots i_0$ where $\overline{i}_b$ is the complement of i_b and $0 \le b < q$. Hence, if i has the binary representation 10110, then $i^{(2)}$ has the representation 10010 and $i^{(0)}$ has the representation 10111. In a cube connected computer, PE(i) is connected to PE($i^{(b)}$), $0 \le b < q$. Figure 2(b) shows the PE interconnections in an 8 PE CCC.

It is important to note that PEs can communicate only via the interconnection network. Besides the mesh and cube connections, several other connections schemes are possible. The reader is referred to Siegel ([SIEG79a] and [SIEG79b]) for a survey of interconnections networks for SIMD computers. The largest SIMD computer currently under construction is the massively parallel processor (MPP) being build by Goodyear Aerospace Research [BATC80]. This machine uses the mesh interconnection scheme



Boxes represent PEs

(a) 4 x 4 MCC



(b) 8 PE CCC

Figure 2

(together with some variations) and will have 16,384 PEs.

An important problem that arises in SIMD computers is that of data routing; moving data from one PE to another. While there are several forms of the data routing problem [NASS81b], we shall deal only with the permutation form. In a permutation problem, PE(i) wishes to send data to PE(A(i)), $0 \le i < N$ where $[A(0),\ldots,A(N-1)]$ is a permutation of $[0,1,\ldots,N-1]$. Arbitrary data permutations are generally accomplished by sorting. For certain classes of permutations, however, their exist algorithms that are more efficient than sorting [NASS81a]. One such class is the BPC (bit permute complement) class of permutations introduced in [NASS80]. A permutation A is a BPC permutation iff it can be described by a vector $B = [B_{q-1}, B_{q-2},\ldots,B_0]$ (where $N = 2^q$ is the number of PEs), such that:

(a) $B_i \in \{\pm 0, \pm 1,\ldots,\pm(q-1)\}$, $0 \le i < q$, and

(b) $[|B_{q-1}|, |B_{q-2}|,\ldots,|B_0|]$ is a permutation of $[0,1,\ldots,q-1]$

The destination d of the data in PE(i) can be computed from this vector B as follows. Let $i_{q-1},\ldots,i_0$ be the binary representation of i and let $d_{q-1},\ldots,d_0$ be the binary representation of d. For $j = 0, 1,\ldots,q-1$, we have:

$$d_{|B_j|} = \begin{cases} i_j & \text{if } B_j \ge 0 \\ \overline{i_j} & \text{if } B_j < 0 \end{cases}$$

Note that we distinguish between +0 and -0 and that $-0 < +0 = 0$. Also, note that the total number of permutations that can be specified in this way is $2^q q! = N(\log N)!$

Intuitively, for each BPC permutation A specified by B, the destination PE for PE(i) is obtained by permuting the bits in the binary representation of i and complementing certain bits. The vector B specifies how the bits are to be permuted and also which bits are to be complemented. $|B_j|$ tells us where bit j is to go, and the sign of B_j tells us if the jth bit of i is to get complemented.

As an example, consider the case N = 16, q = 4 and B = [-0,3,-1,-2]. The data from PE(i), $i = i_3 i_2 i_1 i_0$, is to be routed to PE(j) where $j = j_3 j_2 j_1 j_0 = i_2 i_0 \bar{i}_1 \bar{i}_3$, $0 \le i < N$. The BPC class of permutations includes most of the permutations that commonly arise. Table 1 gives the B vectors corresponding to several popular permutations.

Permutation	Vector Representation
Matrix Transpose	[q/2 - 1,...,0,q - 1,...,q/2]
Bit Reversal	[0,1,2,...,q - 1]
Vector Reversal	[-(q - 1),-(q - 2),...,-0]
Perfect Shuffle	[0,q - 1,q - 2,...,1]
Unshuffle	[q - 2,q - 3,...,0,q - 1]
Shuffled Row Major	[q - 1,q/2 - 1,q - 2,q/2 - 2,...,q/2,0]
Bit Shuffle	[q - 1,q - 3,...,1,q - 2,q - 4,...,0]

Table 1 Some common permutations

Nassimi and Sahni [NASS80] present an optimal algorithm for routing BPC permutations on mesh connected computers. In [NASS81a], they show how BPC permutations may be performed efficiently on a CCC. There algorithm is, however, suboptimal in the sense that for some BPC permutations more data movement may take place than necessary. In this paper, we develop an optimal algorithm for routing BPC permutations on a CCC.

2. Optimal BPC Algorithm

Let $b \in [0,q-1]$. In a unit-route (on a CCC), data can be moved from PE(i) to PE($i^{(b)}$), $0 \le i < N$. Let $B = [B_{q-1},...,B_0]$ be the vector representation of the BPC permutation A = [A(0),...,A(N-1)]. We first obtain a lower bound, $\beta(B)$, on the number of unit-routes needed to perform A on an N PE CCC.

Theorem 1: Let $B = [B_{q-1},\ldots,B_0]$ define the BPC permutation $A = [A(0),\ldots, A(N-1)]$. $\beta(B)$ as given below is a lower bound on the number of unit-routes needed to perform A on a CCC.

$$\beta(B) = |\{b|B_b \neq b\}|$$

Proof: For each b for which $B_b \neq b$, there exists at least one A(i) with the property that $i_b \neq (A(i))_b$ ($(A(i))_b$ denotes bit b of A(i)). Thus, at least one unit-route along bit b is needed. So, at least $\beta(B)$ unit-routes are needed to perform A. $\square$

By making minor modifications to the routing algorithm presented in [NASS81a], we can arrive at an algorithm that performs each BPC permutation B using $2\beta(B) - 1$ unit-routes. The algorithm we are about to present will use exactly $\beta(B)$ unit-routes and is therefore optimal.

Our algorithm follows the cycles present in the bit permutation B. If $(k_1,k_2,\ldots,k_p)$ are the bits in a cycle of A then our algorithm first routes all data to PEs having the correct final value for bit k_1 (i.e. following this route the destination D_i for the data in PE(i) is such that $(D_i)_{k_1} = (i)_{k_1}$). Next, we route along k_2, then k_3, etc. Having finished with this cycle, the next permutation cycle is followed and so on.

Let us first consider an example. Consider performing a perfect shuffle on a CCC with 8 PEs. $B = [0,2,1]$ and the destination A(i) for the data in PE(i) has the binary representation $i_1 i_0 i_2$ (note that the binary representation of i is $i_2 i_1 i_0$). The elements to be permuted are assumed to be in register R of each PE. B has only one cycle $(1,2,0) = (B_0,B_1,B_2)$. The first route is along bit 1, then along bit j, the route is done only for those PEs containing data with destination A(i) such that $i_j \neq (A(i))_j$. In our example, when routing along bit 1, we need to route only data from PE(i) with $i_1 \neq i_0$. This is so because $(A(i))_1 = i_0$ and if $i_1 = i_0$ then the data in PE(i) is already in a PE with the right bit i_1. Data to be routed is moved to a routing register S, and the route performed.

We shall use the following notation and assumptions in specifying our permutation algorithm:

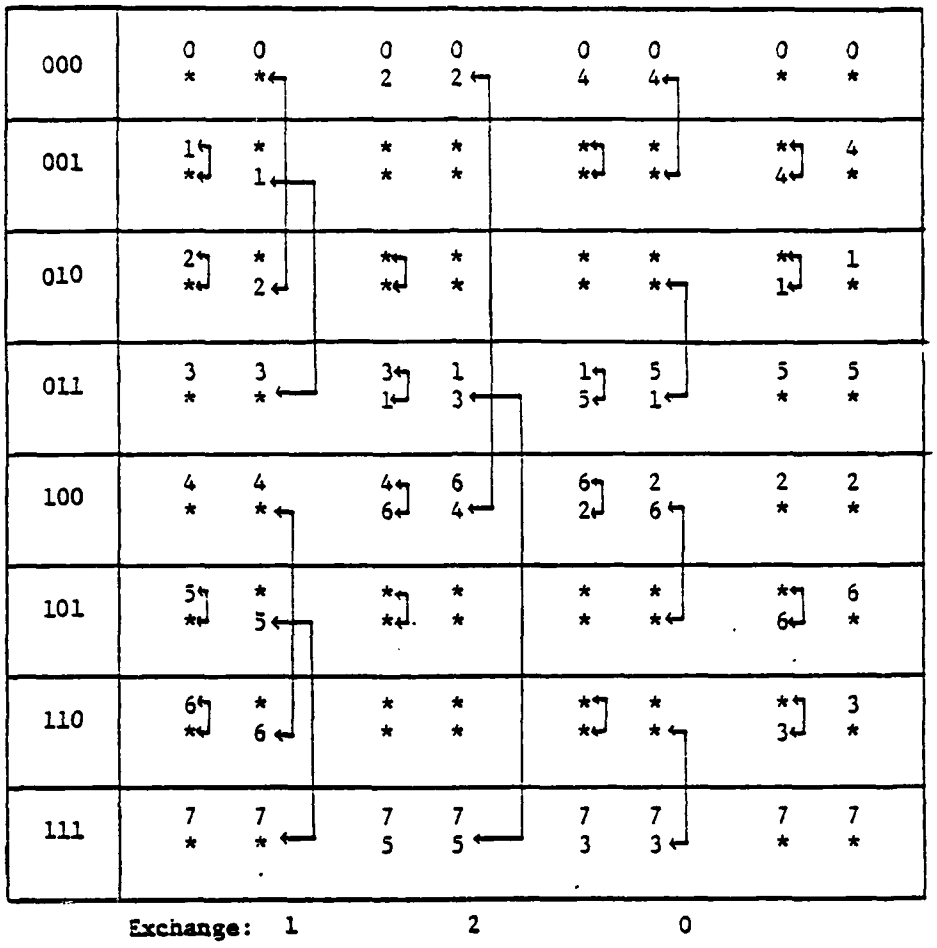


Figure 3: Perfect Shuffle on a Cube

(1) Each PE has two registers R and S. Both these registers are large enough to hold the data being routed. R(i) and S(i) refer to the corresponding registers in PE(i).

(2) Three types of assignments will be used:

(a) := will be used for assignments requiring no routing. For example, R(i) := S(i) (both R and S are in the same PE).

(b) :=: will be used for exchanges requiring no routing. R(i) :=: S(i) results in the R and S registers of PE(i) interchanging data.

(c) ← will denote an assignment requiring a route. We shall require that the PEs denoted by the left and right hand sides be connected by a direct link in the PE interconnection pattern. For example,

$R(i^{(b)}) \leftarrow S(i)$ is valid for a CCC (recall that $i^{(b)}$ is obtained from i by complementing bit b in the binary representation of i). Each assignment of this type is a unit-route.

(3) i_b will denote bit b in the binary representation of i.

(4) PE selectivity can be done using a mask. The mask is specified in parenthesis following the statement. Some examples of masks are:

(i) $(i_b = 1)$: this enables all PEs for which the binary representation of the PE index has bit b equal to 1.

(ii) $(i_j \neq i_k)$: this enables all PEs for which the jth bit in the binary representation of the PE index is different from the kth bit.

When no mask is specified, all PEs are enabled. Instructions are executed only on enabled PEs.

Procedure BPC-CUBE (Figure 4) is a formal statement of our BPC permutation algorithm for CCCs. The loop of lines 1-16 searches for the beginning of a bit cycle. If $|B_b| = b$, we have a cycle of length 1. When $B_b = b$, no work needs to be done. When $B_b = -b$, it is necessary to complement along bit b (line 4). If $|B_b| \neq b$, then we are at the start of a cycle of length more than 1. Lines 7 to 12 follow this cycle. j is used to move along b, $|B_b|$, $|B_{|B_b|}|$, etc. Line 9 puts into S(i) the data to be routed out of PE(i). Line 10 carries out the route along bit k and then, in line 11, B_j is set to j to signify that the cycle containing j will have been taken care of by the time we exit from the case statement. j is moved to the next point on the cycle. Line 14 moves all valid data to the R registers.

We need to elaborate upon two of the statements just made concerning the algorithm. First, we need to show that line 9 moves into the S registers all records that are in a PE whose kth bit does not agree with the kth bit of its destination PE. Secondly, we need to show that line 14 correctly leaves all records in the R registers.

Let TR(i) and TS(i), respectively, denote the source or originating PE for the records currently in R(i) and S(i). Let AR(i) and AS(i) denote the destination PEs for the records in R(i) and S(i), respectively. At the start of each bit cycle (line 6) all records are in R(i). Let $TS(i) = \phi$ and $AS(i) = \phi$. So, initially at line 8 we have (for any cycle): $(TR(i))_j = i_j$ and $(TS(i))_j \neq i_j$, $0 \leq i < N$. (We shall assume that all relations involving

ϕ are true. So, $(\phi)_j = i_j$ is true and $\phi_j \neq i_j$ is also always true.) The relation holds since no routing on bit j could have been performed on any previous iteration of the **for** loop of lines 1 to 16. If $B_j \geq 0$ then for each PE(i) with $i_j \neq i_k$, we have:

$$(AR(i))_k = (TR(i))_j = i_j = \overline{i}_k \text{ and } (AS(i))_k = (TS(i))_j = \overline{i}_j = i_k$$

Hence, R(i) needs to be routed along bit k and S(i) doesn't. If $i_j = i_k$ then we have:

$$(AR(i))_k = (TR(i))_j = i_j = i_k \text{ and } (AS(i))_k = (TS(i))_j = \overline{i}_j = \overline{i}_k$$

Hence, S(i) needs to be routed along bit k and R(i) does not. So, line 9 correctly places into the S registers the records that need to be routed along bit k when $B_j \geq 0$. Using a similar argument one can show the correctness of line 9 when $B_j < 0$. So, following line 11, we have:

$$(AR(i))_k = (AS(i))_k = i_k, \; 0 \leq i < N.$$

```
procedure BPC_CUBE (A,n)
     //Permute R (0:$2^q$-1) according to the BPC permutation B(0:q-1)//
1    for b: = 0 to q-1 do
2         case
3              :$B_b$ = b:   do nothing
4              :$B_b$ = -b:   R($i^{(b)}$) ← R(i)
5              :$|B_b| \neq$ b:
6                   j:=b; s:=$B_b$
7                   repeat
8                        k := $|B_j|$        //Next route is along dimension k//
                         //Put outgoing elements in S//
9                        if $B_j \geq 0$ then S(i) :=: R(i), ($i_j \neq i_k$)
                                      else S(i) :=: R(i), ($i_j = i_k$)
10                       S($i^{(k)}$) ← S(i)
11                       $B_j$:=j; j :=k
12                  until j = b
13                  k := |s|       //s is the initial $B_b$//
14                  if s ≥ 0 then R(i) := S(i), ($i_b \neq i_k$)
                             else R(i):=S(i), ($i_b$=$i_k$)
15        end
16   end
  end BPC_CUBE
```

Figure 4

Also, note that preceding the execution of line 10,

$$(TR(i))_k = (TS(i))_k = i_k$$

as no routes along bit k have yet been performed. Line 10 routes only S. values, so after line 10 we shall have:

$$(TR(i))_k = i_k \text{ and } (TS(i))_k \neq i_k,\ 0 \leq i < N.$$

As a result, on all subsequent iterations of the loop of lines 7-12, we shall have $[(TR(i))_j = i_j$ and $(TS(i))_j \neq i_j]$, $0 \leq i < N$, at line 8. So, line 9 will correctly set S(i) and line 10 will route correctly.

The preceding argument shows that when the loop of lines 7-12 is completed for any b then:

$$(AR(i))_q = (AS(i))_q = i_q$$

for all $q \in \{|B_b|B_{|B_b|}|, \ldots, b\}$.

It remains to move all records back into the R registers (line 14). When a cycle is finished, half the records will be in the R registers and half in the S registers. The records in the S registers need to be moved to the R registers. The first time line 9 is executed for any cycle, $j = b$ and $k = |s|$. When line 10 is executed, records leave half the PEs and the remaining half contain two records each. The empty PEs are those with $i_b \neq i_k$ if $B_b > 0$ and $i_b = i_k$ if $B_b < 0$. Since bits b and B_b do not get used in line 10 again until the last iteration of the repeat-until loop, these PEs remain empty. They get a record only after the last execution of line 10 for this cycle. At this time $k = b$. Thus, the PEs containing records in their S registers are those with index i such that $i_b \neq i_{|s|}$ if $s \geq 0$ and $i_b = i_{|s|}$ if $s < 0$ (note that $s = B_b$). Hence, lines 6 to 14 correctly handle cycles of length more than 1 and leave all records in the R registers. From this and lines 3 and 4, it follows that all cycles are handled correctly and BPC_CUBE performs every BPC permutation. The time complexity of BPC_CUBE is O(log N) and the number of unit-routes (lines 4 and 10) is $\beta(A)$. Hence, BPC_CUBE is optimal.

3. Conclusions

We have presented an optimal BPC routing algorithm for cube connected computers. Several open problems remain. Is their a similarly optimal algorithm to perform BPC permutations on perfect shuffle computers (see [NASS81] for a description of the interconnection network used here)? Can we develop optimal algorithms for other classes of permutations such as omega and inverse omega permutations ([LAWR75]) etc.?

References

[BATC80] K.E. Batcher, "Architecture of a massively parallel processor," 7th Annual Symposium On Computer Architecture, 1980, pp. 168-173.

[LAWR75] D. Lawrie, "Access and alignment of data in array processors," IEEE Trans. On Computers, C-24, 12, pp. 1145-1155, 1975.

[NASS80] D. Nassimi and S. Sahni, "An optimal routing algorithm for mesh-connected computers," JACM, 27, 1, pp. 6-29, 1980.

[NASS81a] D. Nassimi and S. Sahni, "A self routing Benes network and parallel permutation algorithms," IEEE Trans. On Computers, 1981, to appear.

[NASS81b] D. Nassimi and S. Sahni, "Data routing in SIMD computers," to appear.

[SIEG79a] H.J. Siegel, "A model of SIMD machines and a comparison of various interconnection networks," IEEE Trans. On Computers, C-28, 1979, pp. 907-917.

[SIEG79b] H.J. Siegel, "Interconnection networks for SIMD computers," IEEE Computer, vol. 12, no. 16, pp. 57-65, 1979.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO. AD-A106802	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Optimal BPC Permutations On A Cube Connected SIMD Computer		5. TYPE OF REPORT & PERIOD COVERED Technical Report August 1981
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) David Nassimi and Sartaj Sahni		8. CONTRACT OR GRANT NUMBER(s) N00014-80-C-0650 ✓
9. PERFORMING ORGANIZATION NAME AND ADDRESS Computer Science Department University of Minnesota 136 Lind Hall, 207 Church St.,SE, Mpls.,MN 55455		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Department of the Navy Office of Naval Research Arlington VA 22217		12. REPORT DATE August 1981
		13. NUMBER OF PAGES 12
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

BPC permutation, cube connected SIMD computer, complexity.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

In this paper, we develop an algorithm to perform BPC permutations on a cube connected SIMD computer. The class of BPC permuations includes many of the frequently occurring permutations such as matrix transpose, vector reversal, bit shuffle, and perfect shuffle. Our algorithm is shown to be optimal in the sense that it uses the fewest possible number of unit routes to accomplish any BPC permutation.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

FILME
2